计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications [doi: 10.15888/j.cnki.csa.009311]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向异质图的在线图划分算法①

赵新朋, 罗雄飞, 陈楚依, 鄢宝彤, 乔　颖

(中国科学院 软件研究所, 北京 100190)
通信作者: 乔　颖, E-mail: qiaoying@iscas.ac.cn

摘　要: 图划分算法是分布式图计算系统里的重要组成部分, 它将一个图划分为若干子图以便在分布式系统中运行, 并将子图上的点和边数据及子图上的计算任务分配到各分区. 异质图是现实世界中广泛存在的一种图, 它是指具有多种节点类型或边类型的图, 在针对异质图的计算过程中, 现有的图划分算法对于异质图的处理没有考虑到以下问题: 在图计算过程中, 不同类型的节点和边携带的数据量可能不同; 不同的节点和边类型, 可能会采用不同的处理算法, 其计算时间也会不同. 针对现有图划分方法的不足, 本文提出一种面向异质图的在线图划分算法 OGP-HG 算法, 并对现有的 GraphX 图计算引擎进行改进, 将 OGP-HG 算法在改进后的图计算引擎中实现. 本文提出的 OGP-HG 算法通过计算节点划分到不同分区上的负载均衡得分和边划分到不同分区上的数据均衡得分, 得到使异质图负载和内存占用均衡的划分结果. 实验表明, 与传统图划分算法相比, 该算法提高异质图计算效率 1.05–1.4 倍.

关键词: 异质图; 图计算; 图划分; 负载均衡; 内存优化

引用格式: 赵新朋,罗雄飞,陈楚依,鄢宝彤,乔颖.面向异质图的在线图划分算法.计算机系统应用. http://www.c-s-a.org.cn/1003-3254/9311.html

Online Graph Partitioning Algorithm for Heterogeneous Graphs

ZHAO Xin-Peng, LUO Xiong-Fei, CHEN Chu-Yi, YAN Bao-Tong, QIAO Ying

(Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Graph partitioning algorithm is part of distributed graph computing system. It divides a graph into several subgraphs to run in the distributed system and assigns the vertex and edge data and computing tasks on the subgraphs to each partition. Heterogeneous graph is a kind of graph widely existing in the real world. It is a graph with multiple vertex types or edge types. During the calculation of heterogeneous graphs, the existing graph partition algorithms do not consider the following problems. In graph calculation, different vertex and edge types may carry different data amounts. Meanwhile, different vertex and edge types may adopt different processing algorithms, with various calculation time. Aiming at the shortcomings of the existing graph partitioning methods, this study proposes an online graph partitioning algorithm for heterogeneous graphs, OGP-HG algorithm. Additionally, the existing GraphX graph computing engine is improved to implement the proposed algorithm in the improved graph computing engine. The proposed OGP-HG algorithm calculates the load balance score of vertices divided into different partitions and the data balance score of edges divided into different partitions, thus obtaining the division results of balancing the load and memory occupation of heterogeneous graphs. Experiments show that compared with traditional graph partitioning algorithms, this algorithm improves the computing efficiency of heterogeneous graphs by 1.05–1.4 times.

Key words: heterogeneous graph; graph computing; graph partitioning; load balance; memory optimization

① 基金项目: 中国科学院战略先导 C 类课题 (XDC02030300)
收稿时间: 2023-05-10; 修改时间: 2023-06-14; 采用时间: 2023-07-03; csa 在线出版时间: 2023-10-19

图是一种重要的数据结构, 由于以往的关系型数据本身存在拓扑结构、查询效率等问题, 而图数据具有更强大的表达能力, 且可以将多种来源、多种类型的数据合并到一个图数据结构中进行分析, 并获得原本难以获得的分析结果, 因此, 图计算可以广泛地应用在社交网络[1]、推荐系统[2]、网络安全[3]、文本检索和生物医疗等领域, 很多问题能在图论的支撑下使用图算法实现更高效的解决方式. 近年来, 随着大数据、数据挖掘、机器学习等领域技术的高速发展, 很多领域中的图规模变得异常庞大, 另外再加上自然图往往表现出非常倾斜的幂律分布 power-law[4] 特性, 给图计算带来了巨大挑战. 为了应对海量图数据对计算性能的挑战, 分布式图计算系统已被广泛应用. 如何设计高效的图划分算法将大规模图划分为若干子图, 并将子图上的点和边数据及子图上的计算任务分配到多个节点上并行执行, 已成为分布式图计算系统研究中的热点问题.

异质图 (heterogeneous graph) 是指具有多种节点类型或边类型的图数据结构, 能够将现实世界中相互作用且类型各异的网络化数据建模为异质信息网络 (heterogeneous information network, HIN)[5]. 由于域名系统 (domain name system, DNS) 的正常运转需要大量的异构组件配合完成, 为了准确的描述域名系统中各组件之间的关联性和依赖性, 利用异质信息网络能够将其中相互作用的组件包括域名、IP 地址以及逐级授权的域名服务器等, 完整的抽象为图结构[6]. 通过对域名依赖网络的构建, 研究人员能够更好地理解域名系统的复杂性和演化规律, 实现对域名依赖关系的准确分析. 此外, 可基于域名系统中隐藏的丰富语义, 将域名解析场景抽象为的异质图[7], 以挖掘域名解析间的全局关联性. 由于攻击者无法轻易伪造域名系统中域之间逐级授权的关系, 因此异质信息网络能够通过域名的特征分析域名之间的关联, 有利于恶意域名检测的研究[8–10]. 利用 DNS 流量图来表示 DNS 查询序列, 可以检测受感染的客户端、恶意域名以及恶意 DNS 活动[11, 12].

现有的图划分算法都是基于同质图提出的, 对于异质图的处理没有考虑到以下问题: 1) 异质图中不同类型的节点和边所存储的数据占用存储空间大小可能不同; 2) 对于不同类型的节点和边可能会采用不同的处理算法, 其计算时间不同.

主流图计算引擎 GraphX[13]、PowerGraph[14] 等通常采用哈希算法、Grid 算法[15] 或贪婪算法进行图划分, 这些图划分算法基于 GAS 模型[16]、BSP 模型[17] 和点划分模型. 其中, 哈希算法将各边随机均匀地分配到各分区上, 可以使图计算过程中分区内计算负载均衡, 但完全没有考虑图的拓扑结构, 可能使节点的复制系数过高, 导致分区间同步数据的开销过大. Grid 算法通过限制节点复制系数的上限限制了分区间同步数据的开销上限, 但依然没有关注图的拓扑结构, 可能使大量相邻的边分配到不同的分区上, 导致分区间通信开销过大. PowerGraph 中提出的贪婪算法包括 Coordinated (协同) 和 Oblivious (遗忘) 两种实现方式, 这个方法考虑到了图的拓扑结构, 在考虑计算负载均衡的同时, 尽量将相邻的节点和边分配到相同的分区上. 但该算法没有对度数高的节点与度数低的节点区分处理, 可能导致度数高的节点所在分区负载过大, 从而破坏负载均衡, 降低计算性能.

传统的图计算引擎多是离线计算系统, 这是因为以往的图计算多承担的是非实时的离线分析任务. 但是随着业务应用的发展, 图的规模越来越庞大, 一些在线任务也需要采用图计算模型进行计算[18]. 近年来, 学者们又提出了在线图划分算法来应对在线图数据在划分后的负载倾斜问题. Petroni 等人[19] 提出了 HDRF (high-degree (are) replicated first) 算法, 在节点分配决策中考虑节点度, 度数高的节点优先复制. Chen 等人[20] 提出了混合划分算法 Ginger, 对度数高的节点采用点划分, 对度数低的节点采用边划分, 从而获得负载均衡且复制系数低的效果. 但是, 上述图划分算法均未考虑异质图的计算特点, 忽略了异质图节点所运行任务在处理时间与所占存储空间上的差异, 从导致各节点负载的以及所占存储空间的不平衡, 以及计算过程中所占对异质图进行划分时效果不佳, 从而影响后续图计算的效率.

为了解决目前图划分算法在划分异质图时存在的问题, 本文提出了一种新的面向异质图的在线图划分算法, 该方法可以对异质图的节点和边的负载以及存储空间占用进行评估, 并根据评估结果划分异质图数据, 使图划分达到负载均衡、内存均衡、总吞吐量最大的效果, 进而提升了图计算过程的效率.

本文的贡献主要包括: (1) 提出了一种面向异质图数据的在线图划分算法, 通过计算节点划分到不同分区

上的负载均衡得分和边划分到不同分区上的数据均衡得分, 得到使异质图负载和内存占用均衡的划分结果; (2) 对现有的 GraphX 图计算引擎进行改进, 使其适应在线图计算需求, 并将本文提出的在线图划分算法在改进后的图计算引擎中实现, 进而验证了方法的有效性.

1 基本概念与问题定义

定义 1. 异质图. 异质图是指具有多种节点类型或边类型的图. 异质图G可以表示为:

$$G=(V,E) \tag{1}$$

其中, $V=\{v_1,v_2,v_3,\cdots,v_n\}$表示异质图的节点集合, n为节点的总数. 每个节点v均携带数据, 其节点函数记为$F(V_i)$, 表示该节点对所携带数据进行的处理. 图中可有多种类型的节点, 每一类型节点所携带的数据量与节点函数均有所不同. $E=\{e_1,e_2,e_3,\cdots,e_m\}$, 表示图中的有向边集合, m为边的总数. 每条边$e_i=\{v_{\text{src}},v_{\text{dst}}\}$, 其中$v_{\text{src}}$、$v_{\text{dst}}\in V$每条边上携带边属性数据, 不同类型的边所携带的数据量不同.

图计算系统的计算负载分配和图划分算法密切相关. 现有的图计算系统大多基于 GAS 模型, 其计算负载也分为 3 个阶段产生, 即 Gather 阶段、Apply 阶段和 Scatter 阶段. 其中 Gather 阶段和 Scatter 阶段与边的划分相关, Apply 阶段与点的划分相关.

定义 2. 分区负载. 分区P_i的负载为被分配到该分区的节点和边数据处理完所需要的计算资源, 表达式如下:

$$L_i=(G_i+S_i)\times|e_i|+\sum_{v_j\in P_i}A_j \tag{2}$$

其中, L_i为分区P_i的负载, G_i为 Gather 阶段在一条边上做汇集操作花费的资源, S_i为 Scatter 阶段在一条边上做发送操作花费的资源, $|e_i|$为分区P_i上分配到的边数量, A_j为分配到分区P_i上的第j个节点的节点函数执行需要的计算资源.

为了提高图计算性能, 图划分的结果应该使所有分区上的计算负载尽量均衡, 这一方面需要不同分区上的汇集、分发、节点函数计算等操作的负载均衡, 另一方面由于现在的分布式内存计算系统内存占用会影响图计算的效率, 所以也需要不同的分区上边和节点的副本所占用的内存空间尽量均衡. 接下来我们对问题进行形式化定义, 表 1 为问题定义中用到参数列表.

表 1 参数列表

具体参数	参数记号
节点集合	V
边集合	E
分区集合	P
节点函数运行耗时	T
节点所携带的数据量	S_v
边所携带的数据量	S_e
分区平均计算负载	$\overline{L}$
第i个分区内存空间占用	$\|P_i\|$
分区平均内存空间占用	$\overline{\|P\|}$

定义图$G=(V,E),V=(v_1,\cdots,v_n)$为点集合, $E=(e_1,\cdots,e_m)$为边集合, $P=(p_1,\cdots,p_k)$为分区集合, 根据以上拓扑结构引出的问题模型如下.

点和边以批式数据形式到来, 根据节点的节点函数运行耗时T和数据量S_v、边的数据量S_e, 将节点和边分配到不同的分区上去. 分区P_i的负载为L_i. 所有分区的平均计算负载为:

$$\overline{L}=\frac{\sum_{i=1}^{k}L_i}{k} \tag{3}$$

其中, k为分区数.

所有分区中最大分区负载为:

$$L_{\max}=\max\{L_1,L_2,\cdots,L_k\} \tag{4}$$

分区P_i的内存空间占用由其节点与边所携带的数据量所占用内存空间组成, 为:

$$|P_i|=\sum S_v+\sum S_e,\ \forall v\in P_i,\forall e\in P_i \tag{5}$$

所有分区的平均内存空间占用为:

$$\overline{|P|}=\frac{\sum_{i=1}^{k}|P_i|}{k} \tag{6}$$

所有分区中最大分区内存空间占用为:

$$|P|_{\max}=\max\{|P_1|,|P_2|,\cdots,|P_k|\} \tag{7}$$

则分配节点的目标为: 找到使得$L_{\max}-\overline{L}$最小的分发策略, 分配边的目标为: 找到使得$|P|_{\max}$和$|P|_{\max}-\left|\overline{P}\right|$最小的分发策略.

2 基于异质图计算平衡性的在线图划分算法

2.1 计算不平衡性度量

我们对异质图的计算不平衡性的定义分为两部分: 负载不平衡性和内存占用不平衡性. 这是因为在内存

图计算引擎中, 不光计算节点上的运算负载会影响计算效率, 内存占用也会影响计算效率. 具体到异质图上的节点和边来说, 计算不平衡性可以以如下两部分来度量: 异质图中不同类型的节点上节点函数运行耗时不同、不同类型的节点和边上携带的属性数据所占的内存空间不同. 现在主流分布式图计算系统大多使用BSP 模型, 在图计算的一次迭代中, 一旦某一个分区上的计算比较慢, 就会导致其他分区因为等待同步等待造成计算资源的浪费, 从而导致图计算引擎性能下降.

以往的图划分算法都没有关注过异质图的问题, 即把所有待划分的图都看做同质图, 对不同类型的节点和边采取同样的划分逻辑, 这会引起一系列问题: 1) 如果对节点函数运行耗时不同的节点采取相同的划分逻辑, 会导致不同分区上计算负载不同; 2) 对携带不同数据量节点和边采取相同的划分逻辑, 会导致不同分区上内存占用不同, 而内存占用会影响分布式内存处理系统的计算速度.

(1) 负载不平衡因素

分区的计算负载主要取决于分区上边的数量和所有节点的节点函数复杂度. 本文不考虑计算集群的不平衡性, 即认为集群中每个计算节点的算力相同.

为了根据异质图的计算速度不平衡性进行图划分, 需要在图划分前获得节点的节点函数, 节点的类型是有限的, 每种类型节点有其对应的节点函数, 所以节点函数的运行耗时的取值也是有限的. 所有节点函数的运行耗时可以由用户在节点属性数据中直接给出, 也可以在进行图划分前评估. 计算节点函数的运行耗时需要提前获得不同类型节点的节点函数, 将其在系统中各运行 100 次取用时平均值作为运行耗时 (T), 评估获得的 T 以 (VertexType, T) 键值对形式存储.

(2) 存储不平衡因素

在图计算过程中, 一方面, 我们希望让计算需要的数据一直都在内存中, 不会因内存不足导致访问外存. 另一方面, 在图计算的迭代过程中, 一些计算中间量会被保存在内存中, 所以图计算有可能随着迭代过程内存溢出. 数据不平衡除了有可能导致内存溢出外, 也有可能导致性能的问题.

节点的类型是有限的, 每种类型节点有其对应的数据结构, 所以节点的携带的数据量取值也是有限的, 而本文中设定边的类型可以是无限多种, 所以边携带的数据量取值是无限的.

2.2 算法主要思想

本文中提出 OGP-HG (online graph partitioning for heterogeneous graph) 算法来针对异质图进行在线图划分, 根据现在常用的 GAS 图计算模型, OGP-HG 算法从 3 个方向对异质图划分进行优化: 降低节点的复制系数; 使图计算在 Gather、Apply、Scatter 阶段的计算负载均衡; 使所有分区的数据量在各分区上平衡.

(1) 降低节点的复制系数

基于图划分算法的研究我们发现, 在幂律图中, 聚类系数的分布随着节点度的增加而减小. 这意味着低度节点通常属于非常密集的子图, 而这些子图通过高度节点相互连接.

为了利用这一特性, OGP-HG 算法在划分边数据时要尽量切割高度节点, 并在大量分区上复制这些高度节点的副本, 而对于低度节点则尽量不切割, 使其只存储到一个分区上. 由于幂律图中高度节点的数量非常低, 因此只对高度节点进行切割可以使复制系数减少. 具体实现上来说, OGP-HG 算法可以在每个分区上维护一个带有节点部分度的表, 该表随着图划分的过程同时不断更新. 当一条新边被划分到某分区上后, 这个分区的节点部分度表中相应节点的度值将更新. 节点的部分度并不完全等于这个节点的实际度, 但它通常是一个节点实际度的良好指标, 可以一定程度上代表这个节点的实际度. 而如果维护全局的节点度表, 则图划分过程中需要频繁访问这个全局节点度表, 这会引起大量的内网通信, 导致图划分算法的效率过低.

(2) 使图计算在 Gather、Apply、Scatter 阶段的计算负载均衡

Gather 和 Scatter 阶段是图计算的汇集消息和发送消息阶段, 其计算负载主要与分区上的边数量有关. 为了使图计算在 Gather 和 Scatter 阶段的计算负载均衡, 要使各分区上边的数量大致相等.

Apply 阶段是图计算的节点函数计算阶段, 以往的图划分算法都是基于同质图的, 所有节点的节点函数运行耗时相同, 所以并不关注节点的划分, 多数都是按照输入数据的顺序随机分区. OGP-HG 算法关注异质图, 在每个分区上维护一个带有本分区上所有节点函数运行耗时之和的值. 每次将节点划分到节点函数总耗时最小的分区, 并更新这个分区的节点运行总耗时.

(3) 使所有分区的数据量在各分区上平衡

内存空间占用是影响分布式图计算系统性能的重

要因素, 所以要使各分区的数据量尽量均衡, 避免个别分区内存占用过高导致计算过慢, 使其余分区在同步时等待, 甚至造成内存溢出.

由于大多数图的边数量远大于点数据量, 所以我们主要关注边数据和边上两个节点的副本所占用的空间. 在每个分区上维护已经拥有的节点副本的表, 并维护一个带有本分区上所有节点和边数据量的值. 每次将边划分到总数据量最小的分区, 并更新这个分区的节点副本表和总数据量. 已经在分区上建立副本的节点不会重复在这个分区上建立副本, 也不会重复增加总数据量.

OGP-HG 算法的目标可以表述为: 将异质图上的节点和边依次划分到给定的 k 个分区, 在划分过程中考虑节点的计算负载不同, 节点和边的属性数据量也不同, 在这种划分方式下使每个分区上的计算负载均衡, 内存占用平衡, 并最终使图计算的耗时减少, 达到在线图计算的要求.

根据算法主要思想的分析, 划分一条边时要考虑两方面因素, 一是降低复制系数, 二是考虑这条边和其两端节点的数据量, 划分这条边使各分区的内存占用和边数量尽量均衡.

对于降低复制系数因素, 我们参考 HDRF 算法, 提出划分边的目标函数的第 1 部分, 即复制分数:

$$C_{\mathrm{REP}}^{\mathrm{OGP\text{-}HG}}(v_{\mathrm{src}},v_{\mathrm{dst}},p)=g(v_{\mathrm{src}},p)+g(v_{\mathrm{dst}},p) \tag{8}$$

其中,

$$g(v_{\mathrm{src}},p)=\begin{cases}1+\dfrac{\delta(v_{\mathrm{dst}})}{\delta(v_{\mathrm{src}})+\delta(v_{\mathrm{dst}})}, & \text{当} p\in A(v_{\mathrm{src}})\\ 0, & \text{当} p\notin A(v_{\mathrm{src}})\end{cases} \tag{9}$$

$$g(v_{\mathrm{dst}},p)=\begin{cases}1+\dfrac{\delta(v_{\mathrm{src}})}{\delta(v_{\mathrm{src}})+\delta(v_{\mathrm{dst}})}, & \text{当} p\in A(v_{\mathrm{dst}})\\ 0, & \text{当} p\notin A(v_{\mathrm{dst}})\end{cases} \tag{10}$$

这里p表示计算目标函数的分区, $A(v)$表示已经有节点v副本的分区集合, $\delta(v)$表示节点v在已划分子图上的部分度 (即节点v在已划分的子图上有多少条邻边), $g(v,p)$的意思是将节点的部分度归一化后再加一, 通过计算复制分数可以获得使图划分的复制系数最小的分区.

对于划分边的内存占用和边数量均衡因素, 我们改进了只考虑边数量的 HDRF 算法, 加入了对内存占用均衡的考虑, 提出划分边的目标函数的第 2 部分, 即平衡分数:

$$C_{\mathrm{BAL}}^{\mathrm{OGP\text{-}HG}}(v_{\mathrm{src}},v_{\mathrm{dst}},p)=\lambda\frac{maxsize-|p|}{\varepsilon+maxsize-minsize}+\mu\frac{maxmem-mem(p)}{\xi+maxmem-minmem} \tag{11}$$

平衡分数分为两部分, 其中第 1 部分用来平衡不同分区上边的数量, 参数λ用来控制不同分区上边数量的不平衡程度, 当$\lambda=0$时完全不考虑分区边数量的平衡性, λ越大表示分区的边数量平衡性对于算法的影响程度越高. $maxmem$和$minmem$是指当前所有分区中最大和最小的边数量, $mem(p)$是p分区上已有边的数量. ε为预设常数.

第 2 部分用来平衡不同分区上的内存空间占用, 参数 μ用来控制不同分区上内存占用的不平衡程度, 当 $\mu=0$时完全不考虑分区内存占用的平衡性, μ越大表示分区内存占用的平衡性对于算法的影响程度越高. $maxmem$和$minmem$是指当前所有分区中最大和最小占用存储空间, $mem(p)$是p分区上已有数据占用的存储空间. ξ为预设常数.

综合复制分数和平衡分数, 我们得出划分边数据的总目标函数为:

$$C^{\mathrm{OGP\text{-}HG}}(v_{\mathrm{src}},v_{\mathrm{dst}},p)=C_{\mathrm{REP}}^{\mathrm{OGP-HG}}(v_{\mathrm{src}},v_{\mathrm{dst}},p)+C_{\mathrm{BAL}}^{\mathrm{OGP-HG}}(v_{\mathrm{src}},v_{\mathrm{dst}},p) \tag{12}$$

OGP-HG 算法在计算一个节点应当分配到哪个分区上时, 需要得知这个节点的节点函数运行耗时. OGP-HG 算法在计算一条边应当分配到哪个分区上时, 需要得知这条边和其两端点所携带的数据量. OGP-HG 算法为划分节点和边分别设定了一个目标函数, 在划分每一个节点或边时, 计算这个节点或边在每个分区上的目标函数值, 然后将其划分到使其目标函数值最大的分区上. OGP-HG 算法的核心就是设计这两个目标函数.

根据算法主要思想的分析, 划分一个节点时只用考虑其节点函数的耗时, 以保证各分区上的计算负载均衡, 因此我们提出了划分节点的目标函数:

$$C_{\mathrm{BAL}}^{\mathrm{OGP\text{-}HG}}(v,p)=\frac{maxtime-p_t}{\varepsilon+maxtime-mintime} \tag{13}$$

其中, $maxtime$和$mintime$分别表示所有分区内已分配节点的最大、最小节点函数耗时, p_t表示p分区内已分配节点的节点函数耗时, 遍历所有分区的p_t, 找出使$C_{\mathrm{BAL}}^{\mathrm{OGP\text{-}HG}}(v,p)$取得最大值的$p_t$, 即将节点$v$分配到当前

计算负载最小的p分区. 这样依次分配到来的每一个节点, 可以使节点函数的计算负载在各分区均衡. 每分配一个节点后根据分配结果更新对应分区p的p_t.

2.3 算法流程

使用 OGP-HG 算法对异质图进行在线图划分的具体步骤如下.

(1) 异质图数据分别以点数据和边数据形式到来, 每当划分一条边数据时, 遍历所有分区, 找出使得分$C^{\text{OGP-HG}}(v_{\text{src}},v_{\text{dst}},p)$取得最大值的分区$p$, 记为$p_{\max}$, 即将边分配到$p_{\max}$分区上. 每分配一条边后根据分配结果更新$|p|$、$A(v_{\text{src}})$、$A(v_{\text{dst}})$、$\delta(v_{\text{src}})$、$\delta(v_{\text{dst}})$

(2) 每当划分一个点数据时, 根据节点函数时间复杂度T采用划分点的目标函数计算每个节点在各分区上的得分, 使分配节点后不同分区上的计算负载均衡.

(3) 每分配完成一批数据后, 根据所有分区边数量的最大和最小值更新 *maxsize* 和 *minsize*, 根据所有分区内存占用的最大和最小值更新*maxmem*和*minmem*, 根据所有分区节点函数耗费时间的最大和最小值更新*maxtime*和*mintime*.

(4) 将划分好的节点和边组成图, 开始进行图计算迭代运算. 图计算中节点汇集所有邻居节点和邻边的信息并根据自己的节点函数进行计算, 获得计算结果. 当两次迭代的计算结果之差小于设定的收敛值后, 认为这个节点已经达到收敛, 下次迭代时就不会在这个节点进行计算.

(5) 重复步骤 (1)–步骤 (4), 直到没有新的节点或边到来, 且所有节点都达到收敛, 此时, 图划分结束.

使用 OGP-HG 算法对异质图进行在线图划分的流程图如图 1 所示.

在本文所采用的图计算模型中, 所有计算节点都认为是相等的, 它们可以并行地读取图数据, 然后对图进行分割. 由于每个节点都有可能进行图分割任务, 所以当一个节点开始进行图分割任务时, 它就必须拥有所有需要的信息. OGP-HG 算法使用的信息有: 节点的副本集合$A(v)$、节点的部分度$\delta(v)$以及各分区内存占用的最大值和最小值等. 为了方便讨论, 假设在t时刻, 整个图计算引擎有n个分区, 要划分的图有m个节点, k条边.

因此, 对于 OGP-HG 算法中各状态量的维护包括: (1) 使用二维布尔数组维护$A(v)$, 其中第 1 维代表所有节点, 第 2 维代表所有分区. 所以其空间复杂度在一个分区上为$\mathrm{O}(n\times m)$, 在所有分区上为$\mathrm{O}\left(n^2\times m\right)$; (2) 在每个分区上使用整型数组维护$\delta(v)$, 其空间复杂度在一个分区上为$\mathrm{O}(m)$, 在所有分区上为$\mathrm{O}(n\times m)$; (3) 在每个分区上使用浮点数变量维护边数量的最大值 *maxsize* 和最小值 *minsize*、内存占用的最大值*maxmem*和最小值*minmem*, 其空间复杂度在一个分区上为$\mathrm{O}(1)$, 在所有分区上为$\mathrm{O}(n)$; 综合以上分析, OGP-HG 算法的空间复杂度为$\mathrm{O}\left(n^2\times m\right)$.

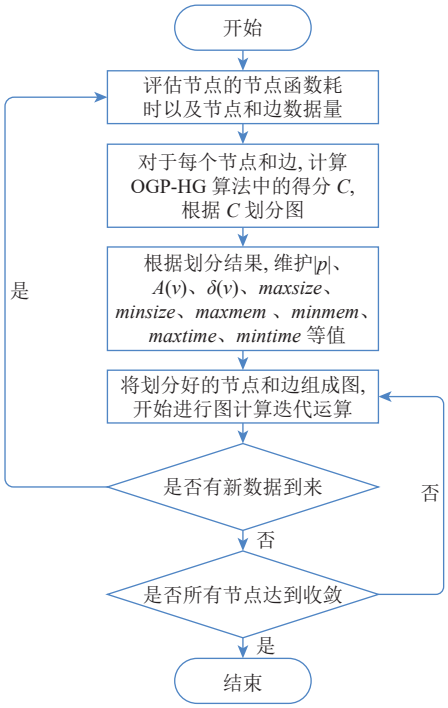


图 1 使用 OGP-HG 算法对异质图进行在线图划分流程图

对于 OGP-HG 算法的计算耗时分为两部分: (1) 计算启发式贪心算法的得分花费的时间, 由于$A(v)$和$\delta(v)$都采取数组存储, 不需要查找, 所以可以直接计算每条边和每个节点在所有分区上的得分, 故这部分的时间复杂度为$\mathrm{O}(n\times(m+k))$; (2) 同步状态量花费的时间, 每次划分一条边后都要同步节点的$A(v)$和$\delta(v)$, 由于每次同步只用同步状态量数组被修改的部分, 不需要全部更新, 故同步的时间复杂度为$\mathrm{O}(n)$; 综合以上分析, OGP-HG 算法的时间复杂度为$\mathrm{O}(n\times(m+k))$.

3 实验与评估

3.1 实验设置

本节使用了 3 个真实数据集对本文中的 OGP-HG 图划分算法和在线图计算引擎进行了实验验证, 真实

数据集分别来自 Twitter 社交网络数据[21]、.uk 域的快照生成图[22]、美国路网图集[23], 这 3 个数据集具有不同的数据规模和拓扑特性. 其中 Twitter 数据集是 2010 年爬取的 Twitter 网上的用户关注网络图, 其数据形式是一系列的边数据, 代表用户间的关注关系, 这一图数据集符合幂律分布; .uk 域的快照生成图是英国 2007 年 5 月的网页快照关系图 (UK2007-05), 图中的节点是当时的网页快照, 边是这些网页的关系, .uk 域的快照生成图符合幂律分布; 美国路网图集 (USA-Road) 展示了美国的道路网络, 图中节点代表不同的地点, 边代表道路, 边上的属性表示物理距离, 这一图数据集不是网络数据集, 不符合幂律分布, 节点的度比较平均, 且普遍度很小.

但是, 本文要验证针对异质图的 OGP-HG 图划分算法的效果, 但是异质图数据集资源很少, 为了验证 OGP-HG 图划分算法对于图计算引擎内存占用均衡的影响, 我们要在这 3 个数据集上做一些修改. 我们自定义一种方法, 将这 3 个数据集上的边和节点都分为两类, 对这些边和节点标记类型, 并对不同类型的节点和边赋予不同数据量的属性数据, 这样就生成了具有多类型节点和边的异质图数据集.

本文一共进行了两组模拟实验. 第 1 组是不同图划分算法下数据均衡性对比实验和不同图划分算法下运行相同图算法耗时对比实验, 旨在检验不同图划分算法下对于实验指标的影响. 对比的图划分算法分别是 GraphX 自带的 Grid 算法、HDRF 算法和本论文中提出的 OGP-HG 算法.

第 2 组实验只使用 OGP-HG 图划分算法, 在图划分结束后进行图计算, 对不同差异度的异质图下的运行效果, 这里的差异度主要指节点函数运行耗时的不同差值倍数. 这一组实验旨在检验本文中提出 OGP-HG 图划分算法对于不同异质程度的图数据会分别产生多大的效果.

为了对比不同节点函数运行耗时差值倍数下的运算耗时, 我们需要运行一个对异质图的不同节点采取不同处理逻辑的图算法. 这里我们将 PageRank 算法做了一个改变, 对于不同类型的节点, 额外加入不同级别的运算负载.

本文实验一共使用了 6 台服务器作为计算节点, 在这 6 台服务器上分布式部署 Spark, 其中一台作为 master 节点, 其余作为 worker 节点. 同时 Spark 读取文件还需要使用 Hadoop 的 HDFS, 也在这 6 台服务器上分布式部署 Hadoop. 服务器参数如表 2.

表 2 硬件参数列表

参数项	参数值
CPU	Inter Core i5
Core	32
Memory	128 GB
Disk	42 TB
OS	Ubuntu 17

软件框架参数如表 3.

表 3 软件参数列表

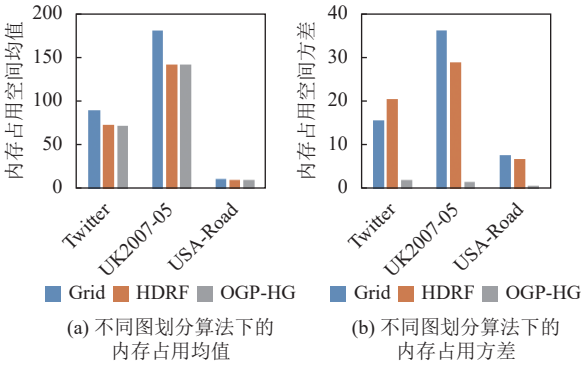
名称	版本
Spark	3.0.3
Zookeeper	3.6.1
JDK	1.8
Hadoop	2.10.0

3.2 评估分析

由图 2(a) 可以看出, 采用 Grid 算法的内存占用均值明显高于另外两个算法, 说明采用 Grid 图划分算法的划分结果复制系数过高, 导致内存占用较高, 而 HDRF 算法和 OGP-HG 算法的内存占用相近. 由图 3(b) 可以看出, 只有 OGP-HG 算法的划分结果内存占用的方差最小, 说明其划分结果的内存占用在各分区上最均衡. 综合两项指标我们得出结论, OGP-HG 图划分算法可以使划分结果的内存占用较小的同时, 又使其内存占用最均衡. 内存占用较小是因为降低了复制系数, 减少了节点的副本. 而内存占用均衡是因为考虑了异质图不同类型和边的内存占用不同, 数据计算耗时将其分配到内存占用最小的分区.

图 3 是对 OGP-HG 与 Grid 和 HDRF 算法在图划分后进行图计算的计算耗时最大值对比, 这一节中采取的图算法为各节点计算负载相差 10 倍的 PageRank 算法. 我们在这里关注计算耗时的最大值, 是因为在线计算通常有时间要求, 只有每一批次的数据计算都能在下一批次数据到来前完成, 才能达到在线计算要求. 观察图 3 可知, OGP-HG 算法相对 HDRF 最高可以节省约 15% 的时间, 且只有 OGP-HG 算法后的图计算可以每一组数据集都在 2 s 内完成, 如果批式数据到来的周期为 2 s, 则只有采用 OGP-HG 算法的在线图计算引擎满足在线要求. OGP-HG 算法耗时最少, 这是因为考虑到了异质图的计算平衡性, 每次分配节点和边时将其分配到负载最小的分区, 最终划分结果各分区计算

负载均衡, 不会出现计算负载多的分区长时间等待的情况.



(a) 不同图划分算法下的内存占用均值　(b) 不同图划分算法下的内存占用方差

图 2　不同图划分算法下内存占用均值和方差

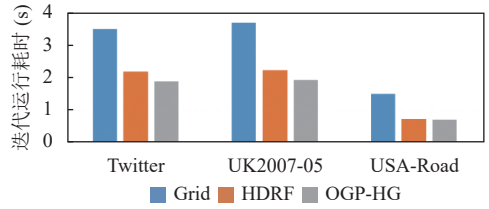


图 3　每一批数据计算耗时最大值

我们验证了 OGP-HG 算法对于异质图计算有较好的效果, 但是对于不同差异程度的异质图, 使用 OGP-HG 图划分算法的效果可能有很大不同. 为了验证这一猜想, 我们将修改后的 PageRank 算法在不同节点上的计算负载分别设定为 1、10、20、30、40、50、60、70、80、90、100, 比较不同节点函数差值倍数的情况下 OGP-HG 算法划分结果对于异质图计算的效果, 以 HDRF 的运行耗时为基准, 实验结果如图 4 所示.

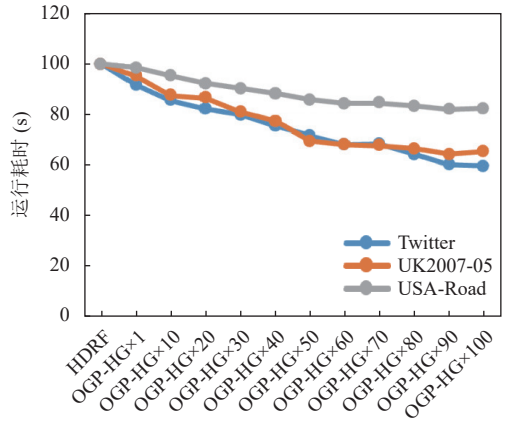


图 4　不同差异程度下异质图计算耗时对比

根据图 4 我们可以得知, 随着异质图不同类型节点的节点函数差异度增大, 使用 OGP-HG 算法后进行异质图计算对比 HDRF 算法节约时间从 5% 到 40% 不等, 而且节点差异程度越大, 使用 OGP-HG 算法后进行图计算的耗时越短, 即图计算的效率越高. 因此我们可以得出结论, OGP-HG 算法对于差异程度越大的异质图的划分效果越好. 这是因为使用以往的图划分算法, 差异程度越大的异质图, 其划分结果负载不均衡的情况就会越突出, 图计算的效率就越差. 相较之下, OGP-HG 算法的效果就会越好.

4　结论与展望

本文在研究异质图计算不平衡性的基础上, 提出了一种面向异质图数据的在线图划分算法, 并对此算法的时间和空间复杂度进行了分析. 为了验证该算法的效果, 本文还对面向异质图数据的在线图划分算法进行了多组实验, 实验结果表明该算法能提升异质图计算性能.

我们可以从以下几个方向展开后续工作: 1) 减少 OGP-HG 算法进行图划分时候的状态量更新开销, 使图划分阶段耗时更小. 2) 目前图神经网络领域有许多异质图方向的研究[24–26], 后续可以以异质图神经网络的计算为案例, 更好地验证本文提出的异质图划分算法的效果.

参考文献

1 Blondel VD, Guillaume JL, Lambiotte R, *et al*. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008: P10008. [doi: 10.1088/1742-5468/2008/10/p10008]

2 Basu C, Hirsh H, Cohen WW, *et al*. Technical paper recommendation: A study in combining multiple information sources. Journal of Artificial Intelligence Research, 2001, 14: 231–252. [doi: 10.1613/jair.739]

3 郑加林, 张志强, 叶安胜. 基于攻击图的网络安全风险计算研究. 信息与电脑, 2015, (19): 176–177. [doi: 10.3969/j.issn.1003-9767.2015.19.075]

4 Calderón F, Curilef S, de Guevara MLL. Probability distribution in a quantitative linguistic problem. Brazilian Journal of Physics, 2009, 39(2A): 500–502. [doi: 10.1590/S0103-97332009000400028]

5 Shi C, Yu PS. Heterogeneous Information Network Analysis and Applications. Cham: Springer, 2017.

6 Guo BY, Zhang M, Xu CX, *et al*. A nameserver importance ranking method based on heterogeneous information

network. Proceedings of the 6th IEEE International Conference on Computer and Communication Systems. Chengdu: IEEE, 2021. 463–468.

7 Sun XQ, Wang ZL, Yang JH, *et al*. Deepdom: Malicious domain detection with scalable and heterogeneous graph convolutional networks. Computers & Security, 2020, 99: 102057.

8 Khalil I, Yu T, Guan B. Discovering malicious domains through passive DNS data graph analysis. Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. Xi'an: ACM, 2016. 663–674.

9 Tran H, Nguyen A, Vo P, *et al*. DNS graph mining for malicious domain detection. Proceedings of the 2017 IEEE International Conference on Big Data. Boston: IEEE, 2017. 4680–4685.

10 Ma Z, Li Q, Meng XY. Discovering suspicious APT families through a large-scale domain graph in information-centric IoT. IEEE Access, 2019, 7: 13917–13926. [doi: 10.1109/ACCESS.2019.2894509]

11 Lee J, Lee H. *GMAD*: Graph-based malware activity detection by DNS traffic analysis. Computer Communications, 2014, 49: 33–47. [doi: 10.1016/j.comcom.2014.04.013]

12 Berger A, D'Alconzo A, Gansterer WN, *et al*. Mining agile DNS traffic using graph analysis for cybercrime detection. Computer Networks, 2016, 100: 28–44. [doi: 10.1016/j.comnet.2016.02.009]

13 Xin RS, Gonzalez JE, Franklin MJ, *et al*. GraphX: A resilient distributed graph system on spark. Proceedings of the 1st International Workshop on Graph Data Management Experiences and Systems. New York: ACM, 2013. 2.

14 Gonzalez JE, Low Y, Gu HJ, *et al*. PowerGraph: Distributed graph-parallel computation on natural graphs. Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. Hollywood: USENIX Association, 2012. 17–30.

15 Jain N, Liao GD, Willke TL. GraphBuilder: Scalable graph ETL framework. Proceedings of the 1st International Workshop on Graph Data Management Experiences and Systems. New York: ACM, 2013. 4.

16 Tian YY, Balmin A, Corsten SA, *et al*. From "think like a vertex" to "think like a graph". Proceedings of the VLDB Endowment, 2013, 7(3): 193–204. [doi: 10.14778/2732232.2732238]

17 Valiant LG. A bridging model for parallel computation. Communications of the ACM, 1990, 33(8): 103–111. [doi: 10.1145/79173.79181]

18 Page L, Brin S, Motwani R, *et al*. The PageRank citation ranking: Bringing order to the web. Proceedings of the 7th International World Wide Web Conference. Brisbane, 1998. 161–172.

19 Petroni F, Querzoni L, Daudjee K, *et al*. HDRF: Stream-based partitioning for power-law graphs. Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. Melbourne: ACM, 2015. 243–252.

20 Chen R, Shi JX, Chen YZ, *et al*. PowerLyra: Differentiated graph computation and partitioning on skewed graphs. ACM Transactions on Parallel Computing, 2019, 5(3): 13.

21 Kwak H, Lee C, Park H, *et al*. What is Twitter, a social network or a news media? Proceedings of the 19th International Conference on World Wide Web. Raleigh North: ACM, 2010. 591–600.

22 Laboratory for Web Algorithmics. UK2007-05. http://law.di.unimi.it/webdata/uk-2007-05/.

23 USA-Road Network. http://users.diag.uniroma1.it/challenge9/download.shtml.

24 Zhang CX, Song DL, Huang C, *et al*. Heterogeneous graph neural network. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Anchorage: ACM, 2019. 793–803.

25 Zhang S, Zhou Z, Li D, *et al*. Attributed heterogeneous graph neural network for malicious domain detection. Proceedings of the 24th IEEE International Conference on Computer Supported Cooperative Work in Design. Dalian: IEEE, 2021. 397–403.

26 Aravind M, Sujadevi VG, Krishnan MR, *et al*. Malicious node identification for DNS data using graph convolutional networks. Proceedings of the 7th IEEE International Conference on Recent Advances and Innovations in Engineering. Mangalore: IEEE, 2022. 104–109.

(校对责编: 牛欣悦)